Amendments to the Specification:

Please replace the paragraph beginning at page 15, line 10, with the following rewritten paragraph:

A1

--**[67]** From each key I provided by dynamic key generator 120, encoder 115 generates an output symbol, with a value B(I), from the input symbols provided by the input symbol generator. The operation of encoder 115 will be described in more detail below. The value of each output symbol is generated based on its key, on some function of one or more of the input symbols, and possibly [on or more]<u>one or more </u>redundant symbols that had been computed from the input symbols. The collection of input symbols and redundant symbols that give rise to a specific output symbol is referred to herein as the output symbol's "associated symbols" or just its "associates". The selection of the function (the "value function") and the associates is done according to a process described in more detail below. Typically, but not always, M is the same for input symbols and output symbols, i.e., they both code for the same number of bits.--

Please replace the paragraph beginning at page 16, line 1, with the following rewritten paragraph:

A2 cont

--**[74]** Fig. 2 is a block diagram of one specific embodiment of encoder 115 shown in Fig. 1. Encoder 115 comprises a static encoder 210, a dynamic encoder 220, and a redundancy calculator 230. Static encoder 210 receives the following inputs: a) original input symbols IS(0), IS(1), ..., IS(K-1) provided by the input symbol generator 110 and stored in an input symbol buffer 205; b) the number K of original input symbols; c) static keys S_0, S_1, ... provided by the static key generator 130; and d) a number R of redundant symbols. Upon receiving these inputs [static encoder 205]<u>static encoder 210</u> computes R redundant symbols RE(0), RE(1), ..., RE(R-1) as will be described below. Typically, but not always, the redundant symbols have the same size as the input symbols. In one specific embodiment, the redundant symbols generated by static encoder 210 are stored in input symbol buffer 205. Input symbol buffer 205 may be only logical, i.e., the file may be physically stored in one place and the positions of the input

A2 concl

symbols within symbol buffer 205 could only be renamings of the positions of these symbols within the original file.--

Please replace the paragraph beginning at page 17, lines 15 and 19, with the following rewritten paragraph:

A3

--**[77]** In situations where the speed of generating output symbols is a critical resource, the input file could be encoded using [static encoder 205]<u>static encoder 210</u> and stored on an intermediate device before the transmission of output symbols starts. This device could be, for example, an attached storage device at a different physical location than dynamic encoder 220, it could be included in the same physical device as dynamic encoder 220, etc. In the case where the file has been encoded with [static encoder 205]<u>static encoder 210</u> well in advance of encoding with dynamic encoder 220, the computational device that implements dynamic encoder 220 need not devote resources to static encoding. Thus, it could devote more resources to dynamic encoding in order to, for example, increase the speed of generating output symbols for the input file, generate output symbols for other files, perform other tasks, etc. Whether or not static encoding can or should be performed in advance of dynamic encoding is dependent upon the particular implementation.--

Please replace the paragraph beginning at page 26, lines 27 and 28, with the following rewritten paragraph:

A4

--**[114]** Fig. 13 is a simplified flow diagram illustrating another embodiment of a method for decoding according to the present invention. This embodiment is similar to that described with respect to [Fig. 11]<u>Fig. 12</u>, and includes steps 1005, 1010, 1015, and [1025]<u>1020</u> in common. But, after step [1025]<u>1020</u>, the flow proceeds to step 1030, in which it is determined whether the input symbols have been recovered to a desired degree of accuracy. If yes, then the flow ends. If no, then the flow proceeds to step 1035. In step 1035, one or more additional output symbols are received. Then, the flow proceeds back to step 1010, so that dynamic decoder 905 and/or static decoder 910 can attempt to recover the remaining unrecovered input symbols.--

Please replace the paragraph beginning at page 30, lines 29, 30 and 31, with the following rewritten paragraph:

A5

--**[129]** Hamming decoder 1210 is also coupled to receive input symbols and redundant symbols from the reconstruction buffer 1215. Additionally, Hamming decoder 1210 receives the number K of input symbols, and the number D, where D+1 is the number of redundant Hamming symbols. Hamming decoder 1210 attempts to recover those input symbols not recovered by the dynamic decoder and the LDPC decoder [2005]<u>1205</u>. While the aim of LDPC decoder [2005]<u>1205</u> is to recover as many as possible input and redundant symbols, Hamming decoder [2010]<u>1210</u> only attempts to recover the input symbols IS(0), IS(1), …, IS(K-1).--

Please replace the paragraph beginning at page 32, line 10, with the following rewritten paragraph:

A6

--**[135]** Dynamic matrix generator 1305 and static matrix generator 1310 will now be described in further detail with reference to dynamic encoder 500 of Fig. 5 and static encoder [205]<u>210</u> in Fig. 2. Fig. 18 is a simplified flow diagram illustrating one embodiment of a method employed by dynamic matrix generator 1305. In step 1405, dynamic matrix generator 1205 initializes a matrix C of format (K+A) x (K+R) to all zeros. Next, in step 1410, the keys I_a, I_b, …. are used in conjunction with weight selector 510 and associator 515 to generate the weights W(0),…,W(K+A-1), and the lists AL(0),…,AL(K+A-1), respectively. Each of the lists AL(k) comprises W(k) integers in the range 0,…,K+R-1. In step 1415, these integers are used to compute C(k,l): Where AL(k)=(a(0),…,a(W(k)-1)), the entries C(k,a(0)),…,C(k,a(W(k)-1)) are set to 1. As explained above, matrix C gives rise to a system of equations for the unknowns (IS(0),...,IS(K-1),RE(0),…,RE(R-1)) in terms of the received symbols (B(0),…,B(K+A-1)). The reason is the following: once dynamic encoder chooses weight W(k) and associate list AL(k)=(a(0),…,a(W(k)-1))), the corresponding output symbol B(k) is obtained as

$$B(k) = L(a(0)) \oplus L(a(1)) \oplus \ldots \oplus L(a(W(k)-1)),$$

wherein L(j) denotes the unknown value of reconstruction buffer 1925 at position j. These equations, accumulated for all values of k between 0 and K+A-1, give rise to the desired system of equations.--

Please replace the paragraph beginning at page 33, line 34 and page 35, line 2, with the following rewritten paragraph:

A1

--**[141]** Another embodiment of a method for implementing an associator 520 for which N need not be a prime number is shown in Fig. 20. First, in a step 1805, a variable k is initialized to zero. Then, in a step 1810, a random integer Y is generated. In one specific embodiment, the key I for the output symbol is used to seed a random number generator. Then, in step 1815, the integer Y is taken modulo the number N to produce a number between 0 and N-1. In step 1820, the candidate number Y is tested against other numbers Y previously generated (X(0), X(1), ...). If the number Y had been previously generated, then the flow returns to step 1810. Otherwise, in step 1825, it is included in a list X(0), X(1). Then, in step 1830, it is determined whether W(I) numbers have been generated. If not, then the flow returns to step 1810. The result of the flow illustrated in Fig. [8]<u>20</u> is a list of W(I) numbers X(0), X(1), ... X(W(I)-1), where each number X in the list is a unique integer between 0 and N-1. Then, in a step [835]<u>1835</u>, the list AL(I) is set as the numbers X(0), X(1), ... X(W(I)-1).--